Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование ассиметричных шифров RSA и Эль-Гамаля

Студент: Коктыш Е. С.

ФИТ 3 курс 6 группа

Преподаватель: Нистюк О. А.

Минск 2024

1. **Теоретические сведения**

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования (открытый или публичный ключ), другой – для расшифрования (тайный ключ). Отметим, что указанная пара ключей принадлежит получателю зашифрованного сообщения.

Все алгоритмы шифрования с открытым ключом основаны на использовании односторонних функций, к числу которых, как известно, относится вычисление дискретного логарифма.

Определение 1. Односторонней функцией (one-way function) называется математическая функция, которую относительно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента, т. е., зная х, легко вычислить f(x), но по известному f(x) трудно найти подходящее значение x.

По мнению Диффи и Хеллмана алгоритм шифрования с открытым ключом, должен:

* вычислительно легко создавать пару (открытый ключ, e – закрытый ключ, d),
* вычислительно легко зашифровывать сообщение Mi открытым ключом,
* вычислительно легко расшифровывать сообщение Ci, используя закрытый ключ,
* обеспечивать непреодолимую вычислительную сложность определения соответствующего закрытого ключа при известном открытом ключе,
* обеспечивать непреодолимую вычислительную сложность восстановления исходного (открытого сообщения, Mi) зная только открытый ключ и зашифрованное сообщение, Ci.

В наиболее известном варианте задачи о ранце требуется выяснить, обладает или нет данный вход (S, S) решением. В варианте, используемом в криптографии, нужно для данного входа (S, S) построить решение, зная, что такое решение существует. Оба эти варианта являются NP-полными. Имеются также варианты этой задачи, которые не лежат даже в классе NP.

Как видим, проблема укладки ранца формулируется просто. Дано множество предметов общим числом n различного веса. Спрашивается, можно ли положить некоторые из этих предметов в ранец так, чтобы его вес стал равен определенному значению S? Более формально задача формулируется так: дан набор значений s1, s2, …, sn и суммарное значение S. Требуется вычислить значения si такие, что

S = b1\*s1 + b2\*s2+... + bn\*sn. (1.1)

Здесь bi может быть либо нулем, либо единицей. Значение bi = 1 означает, что предмет mi кладут в рюкзак, а bi = 0 – не кладут.

Трудный для укладки ранец применяется в качестве открытого ключа е, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа d применяется легкий для укладки ранец, который предоставляет простой способ расшифрования сообщения.

В качестве закрытого ключа d (легкого для укладки ранца) используется сверхвозрастающая последовательность, состоящая из n элементов: d1, d2, …, dz: d = {di}, i = 1, …, n.

1. В качестве текущего выбирается число S, которое сравнивается с «весом» самого тяжелого предмета (dn); если текущий вес меньше веса данного предмета, то его в ранец не кладут (0), в противном случае его укладывают (bn = 1) в ранец и переходят к анализу очередного (в общем случае – i-го предмета).

2. Если на предыдущем (i-м шаге) предмет пополнил ранец, то текущий вес уменьшают на вес положенного предмета (S = S – di); переходят к следующему по весу предмету в последовательности: di-1.

Шаги повторяются до тех пор, пока процесс не закончится.

Для получения открытого ключа e (e = {ei}, i = 1, …, n) все значения закрытого ключа умножаются на некоторое число a по модулю n:

ei = di \* a (mod n). (1.2)

Значение модуля n должно быть больше суммы всех чисел последовательности; кроме того, НОД (а, n) = 1.

Для зашифрования сообщения (М) оно сначала разбивается на блоки, по размерам равные числу (z) элементов последовательности в ранце. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков (Si, i = 1, . . ., z): по одному ранцу для каждого блока сообщения с использованием открытого ключа получателя, e.

Для расшифрования сообщения получатель (использует свой тайный ключ, d: сверхвозрастающую последовательность) должен сначала определить обратное к а число: а-1, такое что:

а \* а-1 (mod n) = 1. (1.3)

Для вычисления обратных чисел по модулю можно использовать известный нам расширенный алгоритм Евклида.

После определения обратного числа каждое значение шифрограммы (ci) преобразуется в соответствии со следующим соотношением:

Si = ci \* а-1 mod n (1.4)

Полученное на основании последней формулы для каждого блока число далее рассматривается как заданный вес ранца, который следует упаковать по изложенному выше алгоритму, используя сверхвозрастающую последовательность (тайный ключ получателя).

1. **Практическая часть**

В данной лабораторной работе необходимо разработать авторское оконное приложение, которое должно реализовывать следующие операции:

• генерация сверхвозрастающей последовательности (тайного ключа);

• вычисление нормальной последовательности (открытого ключа);

• зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;

• расшифрование сообщения;

• оценка времени выполнения операций зашифрования и расшифрования.

**2.1. Табличная форма зависимости времени вычисления**

С помощью консольного приложения составить табличную форму зависимости времени вычисления параметра у, функционально заданного выражением вида:

у ≡ ax mod n,

от параметров a, x, n.

Параметр а – это десятичные числа от 5 до 35. Было выбраны числа: 5, 25. Параметр x – это простые числа из диапазона от 103 до 10100.  Было выбрано 10 чисел: 10007, 20483, 40961, 65537, 131071, 262147, 524287, 1048573, 2088571, 4176901. Параметр n – два числа, в двоичном виде состоящие из 1024 и 2048 битов. Код приложения представлен в листинге 2.1.

|  |
| --- |
| function moduloMultiplication(a, x, n) {  let result = BigInt(1);      let temp = BigInt(a);      while (x > BigInt(0)) {    if (x & BigInt(1)) {          result = (result \* temp) % BigInt(n);    }        temp = (temp \* temp) % BigInt(n);    x >>= BigInt(1);      }      return result;}    function runModuloMultiplication() {      const aValues = [5, 25];    const xValues = [10007, 20483, 40961, 65537, 131071, 262147, 524287, 1048573, 2088571, 4176901];    const nValues = [1024, 2048];      for (const a of aValues) {    for (const x of xValues) {          for (const n of nValues) {            const startTime = new Date().getTime();        const y = moduloMultiplication(a, x, n);            const endTime = new Date().getTime();        const elapsedTime = endTime - startTime;            console.log(`a = ${a}, x = ${x}, n = ${n.toString(2)}, y = ${y}, time = ${elapsedTime} ms`);          }    }      }}  runModuloMultiplication() |

Листинг 2.1 – Функция для табличной формы

 Время вычисления значения y возрастает при увеличении числа x. Результат работы приложения приведен на рисунке 2.2.

|  |
| --- |
|  |

Рисунок 2.1 – Вывод в консоль зависимости вычисления параметра y

**2.2. RSA**

Приложение должно реализовывать следующее:

Шифрование и расшифрование текста на основе алгоритма RSA.

Для генерации двух ключей: тайного и открытого (а по сути – двух взаимосвязанных частей одного ключа, т. е. ключа, принадлежащего одному физическому лицу (или группе лиц), либо одному юридическому лицу), используются два больших случайных простых числа p и q. Для максимальной большей криптостойкости нужно выбирать p и q равной длины. Рассчитывается произведение: n = pq. Это есть один из трех компонент ключа, состоящего из чисел n, e, d.

Затем случайным образом выбирается второй компонент ключа (открытый ключ или ключ зашифрования, e, такой что e и (p – 1)(q – 1) являются взаимно простыми числами; вспомним, что (p – 1)(q – 1) = φ(n) – функция Эйлера). Б. Шнайер [5] рекомендует число е выбирать из ряда: 3, 17, 216 + 1. Наконец, расширенный алгоритм Евклида используется для вычисления третьего компонента ключа: ключа расшифрования d такого, что выполняется условие: ed ≡ 1 (mod φ(n)). (8.4) Другими словами: d–1 ≡ e (mod φ(n)). (8.5)

Таким образом, сформирован ключ, состоящий из трех чисел, которые в свою очередь образуют две вышеупомянутые взаимосвязанные части: открытый (публичный) ключ (e, n) и тайный ключ (d, n; на самом деле, как видим, тайным здесь является лишь первое из пары чисел).

**2.2.1. Шифрование**

Исходный текст для зашифрования – собственные фамилия, имя, отчество.

Функция принимает на вход оригинальный текст и публичный ключ и осуществляет зашифрование с помощью алгоритма RSA. Код функции представлен в листинге 2.2.

|  |
| --- |
| CryptoService.rsaDecrypt = function (data, privateKey) {          var buffer = Buffer.from(data, 'base64');          var decryptedData = crypto\_1.privateDecrypt({ key: privateKey, padding: crypto\_1.constants.RSA\_PKCS1\_OAEP\_PADDING }, buffer);          return decryptedData.toString('utf-8');      }; |

Листинг 2.2 – Код функции

Результат работы приложения с исходным текстом «Koktysh Evgenia Sergeevna» представлен на рисунке 2.2.

|  |
| --- |
|  |

Рисунок 2.2 – Результат шифрования

**2.2.2. Расшифрование**

Для дешифрации сообщения нам необходимо найти число, обратное A по модулю N. После нахождения A-1 нам поэлементно найти остаток от деления произведения числа шифротекста и A-1 по модулю N. Мы получим какое-то число. Это число является суммой элементов изначальной последовательности, на месте которых была единичка в 6- или 8-битном блоке сообщения. Код функции для расшифрования представлен в листинге 2.3.

|  |
| --- |
| CryptoService.rsaEncrypt = function (data, publicKey) {          var buffer = Buffer.from(data);  var encryptedData = crypto\_1.publicEncrypt({ key: publicKey, padding: crypto\_1.constants.RSA\_PKCS1\_OAEP\_PADDING }, buffer);          return encryptedData.toString('base64');      }; |

Листинг 2.3 – Код функции расшифрования

Результат работы приложения представлен на рисунке 2.3.

|  |
| --- |
|  |

Рисунок 2.3 – Результат расшифрования

**2.3. Алгоритмом Эль-Гамаля**

Приложение должно реализовывать следующее:

Шифрование и расшифрование текстовых документов на основе алгоритма алгоритма Эль-Гамаля.

Выбирается простое число р. Выбирается число (g, g < p), являющееся первообразным корнем числа р – очень важный элемент с точки зрения безопасности алгоритма (см. ниже). Далее выбирается число х (х < p) и вычисляется последний компонент ключевой информации: y ≡ gх mod р. (8.8) Владельцу сформированной ключевой информации, состоящей из 4 чисел, может посылаться некоторый шифртекст, созданный с использованием открытого ключа получателя: p, g, y. Расшифрование шифртекста получатель производит своим тайным ключом: p, g, х.

**2.3.1. Шифрование**

Исходный текст для зашифрования – собственные фамилия, имя, отчество.

Функция принимает на вход оригинальный текст и параметры публичного ключа и осуществляет зашифрование с помощью алгоритма Эль-Гамаля. Код функции представлен в листинге 2.4.

|  |
| --- |
| CryptoService.decryptElGamal = function (p, x, encryptedText) {          var result = "";          var arr = encryptedText.split(' ').filter(function (xx) { return xx !== ""; });          for (var i = 0; i < arr.length; i += 2) {              var a = parseInt(arr[i]);              var b = parseInt(arr[i + 1]);              if (a !== 0 && b !== 0) {                  var deM = this.multiplication(b, this.exponentiation(a, p - 1 - x, p), p);                  var m = String.fromCharCode(deM);                  result += m;              }          }          return result;      };      return CryptoService;  }()); |

Листинг 2.4 – Код функции

Результат работы приложения с исходным текстом «Koktysh Evgenia Sergeevna» представлен на рисунке 2.4.

|  |
| --- |
|  |

Рисунок 2.4 – Результат шифрования

**2.3.2. Расшифрование**

Если для зашифрования двух разных блоков (m1 и m2) некоторого сообщения использовать одинаковые k, то для соответствующих шифртекстов c1 = (a1, b1) и c2 = (a2, b2) выполняется соотношение b1(b2) –1 = m1(m2) –1. Из этого выражения можно легко вычислить m2, если известно m1. При примерно одинаковой размерности ключей рассмотренные алгоритмы обеспечивают примерно одинаковый уровень криптостойкости. Код функции для расшифрования представлен в листинге 2.5.

|  |
| --- |
| CryptoService.encryptElGamal = function (p, g, x, originalString) {          var result = "";          var y = this.exponentiation(g, x, p);          for (var \_i = 0, originalString\_1 = originalString; \_i < originalString\_1.length; \_i++) {              var char = originalString\_1[\_i];              var code = char.charCodeAt(0);              if (code > 0) {                  var k = Math.floor(Math.random() \* (p - 2)) + 1;                  var a = this.exponentiation(g, k, p);                  var b = this.multiplication(this.exponentiation(y, k, p), code, p);                  result += "".concat(a, " ").concat(b, " ");              }          }          return result;      }; |

Листинг 2.5 – Код функции расшифрования

Расшифрование с помощью алгоритма Эль-Гамаля заняло меньше времени по сравнению с алгоритмом RSA и составило 10 мс. Результат работы приложения представлен на рисунке 2.5.

|  |
| --- |
|  |

Рисунок 2.5 – Результат расшифрования

# Время выполнения

Чтобы оценить скорость работы и изменение объема криптотекстов, построим графики. На рисунке 3 представлен график зависимости выполнения алгоритмов от количества символов в сообщении.

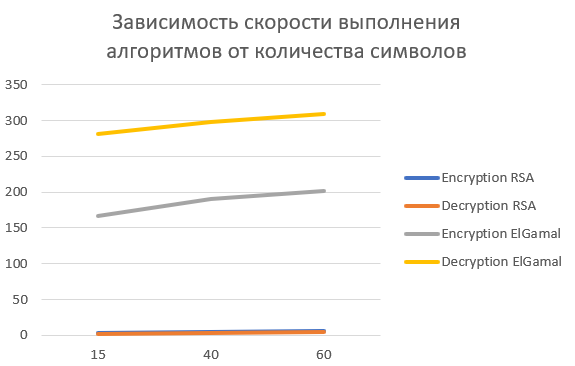


Рисунок 3 — График зависимости скорости выполнения алгоритма, от кол-ва символов в сообщении

На рисунке 3.1 представлена зависимость количества символов от выбранного алгоритма шифрования.

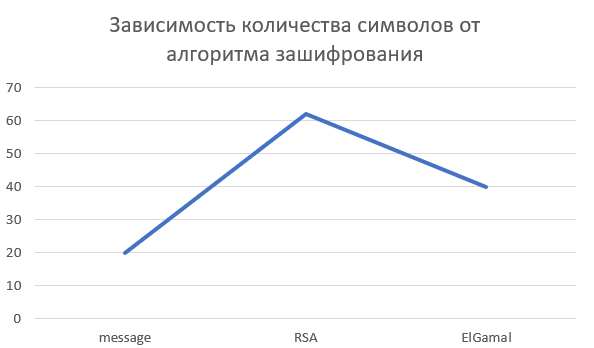


Рисунок 3.1 – Зависимость количества символов от алгоритма шифрования

Можно заметить, что больше всего места занимает шифротекст в RSA, Эль-Гамаля увеличивает количество в 2 раза, поскольку число представляет собой пару чисел.

Как видно из статистики шифра текст увеличился в несколько раз, в особенности если смотреть на алгоритм Эль-Гамаля. Это происходит, из-за использования двух блок в шифр тексте на один блок исходного текста.

# Вывод

В ходе выполнения данной лабораторной работы мы изучили и приобрели практические навыки разработки и использования приложений для реализации ассиметричных шифров RSA и Эль-Гамаля.